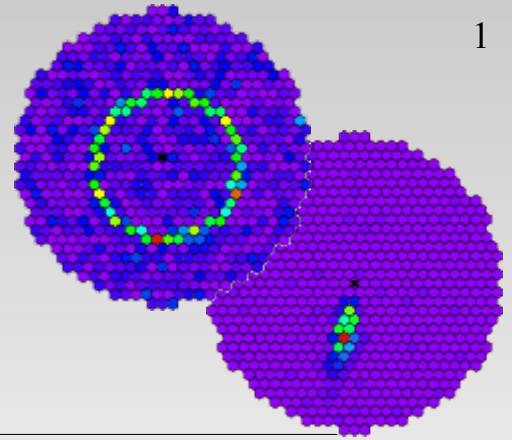
Software requirements

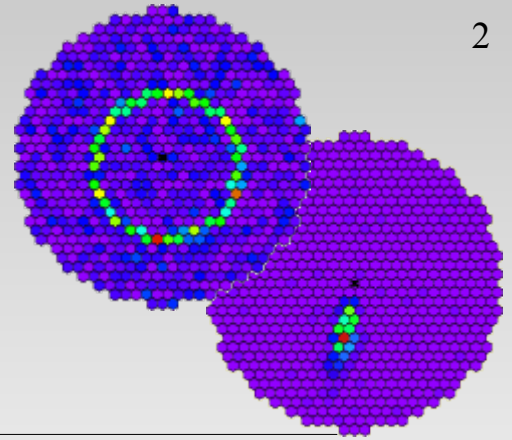
Some general *rules*



- Software written in C++ using root were suitable
- Software should be stored in our cvs
- All code must be well documented!
- The code should follow the MARS standards
- No code should ever be duplicated. Very often root, any other library or MARS has already the necessary function. If the existing functions don't fit your needs let us adapt the existing function and don't copy code!
- Just a few exmaples:
  - Spline interpolation / Signal extraction
  - Observator location
  - (simple) astrometric conversions
  - Camera geometry and display
  - I/O, logging
  - Math functions

Software requirements

Concept



- Note that it is possible to combine root and QT GUIs.
- We will follow a strict policy of releases and version numbers as soon as we start operation. The changes from one release or version to the next must be trackable. Version numbers must be send/written in any stream (UDP, TCP/IP, Files, ...)

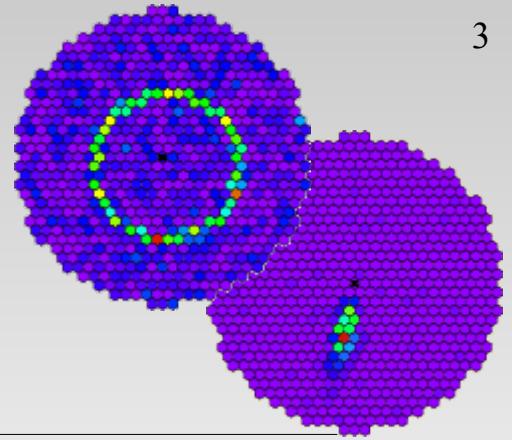
- The goal is to implement all data formats in a way that the least number of changes in MARS is needed.

- Each part of the software should be made as flexible as possible. If we ever build a second DWARF anywhere in the world we don't want to re-write everything, but just change the configuration.

- The code should be as modular as possible to allow easy maintanance (also for non-experts and to keep flexibility)

Software requirements

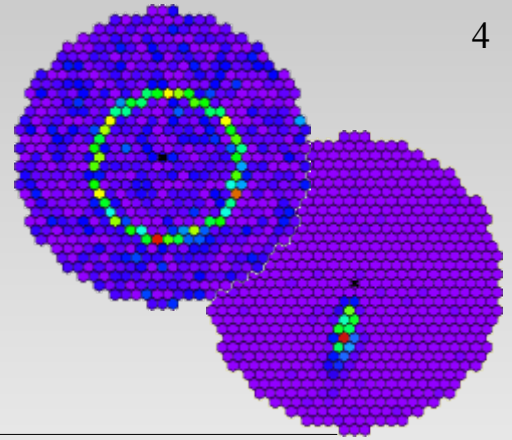
Concept



- The easier a solution/algorithm is the more robust it is usually
- Nobody gets harmed by a useless sanity check! We aim for a stable program.
- All programs should be debugged using valgrind, and easy optimizations should be done using callgrind (e.g. recalculation of sqrt(2) millions of times, ...)

Software requirements

On-line software



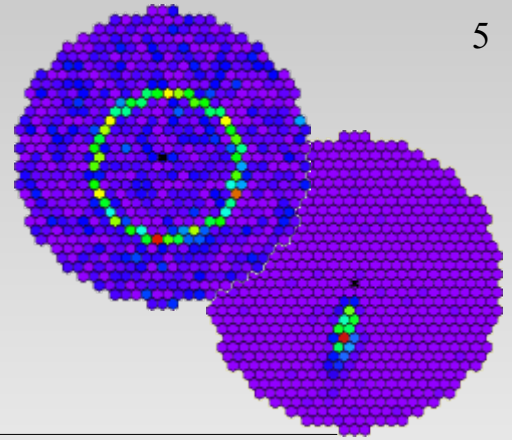
- The core running on La Palma should implement only functions really really necessary.
- Functions not really necessary should be implemented in client programs which only have the absolutely necessary communication with the core
- A guard program should watch its activities and restart it if necessary

- The GUI should be a standalone program. Several GUIs should be able to connect at the same time. To make this stable and possible the GUIs should not connect to the core but a *server* which just redirects the connections from and to the core (the core should not know how many clients are connected) and also check user rights.

- Core and server should communicate via UDP or sockets. Commands should be ackknowledged. Continous status reports might get lost (but we should make sure that not more than n status reports get lost in x seconds)

Software requirements

On-line software



- The core also is responsible for starting and stopping data taking and interacts directly with the control- and run-database.
- The core might also implement the online anlysis (which just needs a proper interface in MARS) or directs the data to a special MARS client.
- It would be nice to have a possibility to check and debug the software in the lab. This could be realized by implementing a fake core which sends previously recorded information.